Applying fuzzy automata to analyze heart data

Trabajo de fin de grado del Doble Grado en Ingeniería Informática - Matemáticas

Iván Calvo Revelo

Department of "Sistemas Informáticos y Computación"
Facultad de Informática
Universidad Complutense de Madrid

June 2018

Document layout with TEXIS v.1.0.

This document is prepared to be printed double-sided.

Applying fuzzy automata to analyze heart data

Trabajo fin de grado
Doble Grado en Ingeniería Informática - Matemáticas
Universidad Complutense de Madrid

Directed by
Mercedes García Merayo
Manuel Núñez García

Department of "Sistemas Informáticos y Computación"
Facultad de Informática
Universidad Complutense de Madrid

June 2018

Acknowledgements

I would like to thank my thesis supervisors for their help and their wise pieces of advice. I would also like to thank TEXIS creators for saving me a lot of time. Finally, I would like to thank my family for their unconditional support.

Abstract

There has been several attempts to introduce formal methods in the development of medical computer systems. Fuzzy automata provide a way to cope with imprecision, which appears in almost every biological or medical system. In this Thesis, we improve and extend a previous formalism, based on fuzzy automata, and develop tools to facilitate the definition of models using our formalism and its practical use. We have applied our formalism and tools to analyze heart data to detect and prevent arrhythmia.

Key Words: Formal methods; Fuzzy automata; Development of software tools.

Resumen

Existen varios intentos de usar los métodos formales en el desarrollo de sistemas informáticos médicos. Los autómatas difusos ofrecen la posibilidad de tener en cuenta la imprecisión, una característica que está presente en casi cualquier sistema médico o biológico. En este trabajo extendemos y adaptamos un formalismo existente, basado en autómatas difusos, y desarrollamos herramientas para facilitar tanto la definición de modelos que usan nuestro formalismo como su uso. Hemos aplicado nuestro formalismo y herramientas al análisis de datos cardiacos para detectar y prevenir arritmias.

Palabras Clave: Métodos formales; Autómatas difusos; Desarrollo de herramientas software.

Contents

List of Figures

Chapter 1

Introduction

In this initial chapter of the Thesis we present a brief introduction to our work. The rest of the chapter is structured as follows. First, in Section 1.1 we present the motivation of our work and review previous work on the topic. In Section 1.2 we briefly describe the goals that we set when we started the work in this Thesis. Finally, in Section 1.3 we present our Work Plan where we provide some key dates and describe the main activities.

1.1 Motivation and previous work

The use of formal methods in the development of complex systems improves their reliability and it has been advocated that they should be used in the same way as blueprints are used in other Engineering fields Lamport (2015). Actually, the use of models as a basis to develop and analyze systems can be found in other scientific fields Magnani y Bertolotti (2017). Unfortunately, there are several obstacles, mainly associated with their complexity and the lack of tools to support them Gogolla (2004); Bjørner y Havelund (2014), to achieve a widespread use of formal methods. In addition, general purpose formalisms are not suitable to be used in specific fields. This is the case of the main application area considered in this Thesis: modeling and analyzing the behavior of the heart.

There are several approaches to formally model the heart Hunter et al. (2003); Jiang et al. (2010); Méry y Singh (2012); Chen et al. (2014) but they fail to take into account common characteristics in biological systems such as uncertainty and imprecision. If we use inaccurate models to analyze a system (whether biological or not), then we will not be able to obtain useful results. Fuzzy logic Zadeh (1965, 1996) has a well established mathematical theory and it can be appropriately used to model real systems because it allows us to use a degree of *imprecision*. Actually, the literature has several proposals to *fuzzify* automata Wee y Fu (1969); Mraz et al. (1999); Mordeson y Malik (2002); Doostfatemeh y Kremer (2005). Moreover, it is worth

noting that fuzzy logic has been previously used to model the behavior of the heart Wojtasik et al. (2004); Dorn (2007).

This thesis takes as starting point a variant of fuzzy automata Andrés et al. (2011) oriented to the specification and testing of systems with uncertain information about time. This initial formalism has been slightly reformulated since its initial definition and used in different application areas Boubeta-Puig et al. (2017a,b); Camacho et al. (2017). This recent work has shown some of its weaknesses, in particular, while modeling and analyzing information about the heart.

1.2 Goals of our work

The main goal before we started the work in this Thesis was to introduce a new formalism that can be used to formally specify complex systems where uncertainty plays an important role. We wanted to provide an improved version of the previously mentioned formalisms, *fuzzy* versions of finite automata, and define its syntax and semantics.

Strongly related to the first goal, a second goal of the work was to apply the formalism to a non-trivial case study. Building on top of previous work developed in the research group, we wanted to apply our formalism to define and analyze information extracted from ECGs (electrocardiograms). Actually, it is specially interesting to model the behavior of the heart by taking into account some specific data extracted from ECGs such as BPM (heartbeats per minute) and RR intervals (interval between two consecutive R waves in the electrocardiogram). In this line, we considered normal values of ECGs as provided by the study of numerous patients Rijnbeek et al. (2014); Haarmark et al. (2010). In order to assess the usefulness of the model, we should use existing databases, including information of real patients, to check whether our model detects existing illnesses.

The last goal of our work was to provide a tool supporting our formal framework. Once we had a running version of our tool we should use it to review our previous model and analyze, again, heart data to detect anomalies associated with arrythmia.

1.3 Work Plan

There are two major lines of work that we considered during the development of this Thesis. First, we have a *theoretical line of work* where we defined the syntax and (operational) semantics of our formalism and analyzed a case study. This language is a revised and improved version of a previous language Camacho et al. (2017) in which the supervisors of this Thesis were heavily involved. The work in the first part of the Thesis started in the

beginning of July 2017 and finished in mid October 2017. The results of our work were compiled in a research paper and submitted to the *10th Asian Conference on Intelligent Information and Database Systems*, ACIIDS 2018. The paper was accepted Calvo et al. (2018a). We continued polishing the formalism and the code used in the case study until we submitted the camera-ready of the paper in mid December 2017. Chapter 2 of this Thesis includes an extended version of our paper Calvo et al. (2018a) including, in particular, some of the procedures used to compute the results. The Python scripts used to compute these results are available at `https://github.com/FINDOSKDI/heartdiagnosis`.

The end of the first phase, including a complete revision of the code used in our experiments, led us to the beginning of the second major line of work, a more practical one, considered in this Thesis. This line of work consisted in the development of `AUNTY`: a tool to AUtomatically aNalyze daTa using fuzzY automata. We were able to have an operational version of our tool by the end of February 2018 and prepared a submission to the *3rd International Conference on Computational Intelligence and Applications*, ICIIA 2018. The paper was accepted Calvo et al. (2018b). Chapter 3 takes as starting point our paper but it includes new material such as the formal syntax of the internal languages used in the representation of models based on our formalism. The Python code of `AUNTY` is available at `https://github.com/FINDOSKDI/AUNTY`.

The following table contains the main information related to our Work Plan (some dates are approximate):

Initial date	**End date**	**Description of the work**
July 1, 2017	Sep. 30, 2017	Study of previous formalisms and definition of new one
Oct. 1, 2017	Oct. 15, 2017	Preparation of submission to ACIIDS 2018
Nov. 15, 2017	Dec. 15, 2017	Polish formalism and improve code; towards `AUNTY`
Feb. 1, 2018	Feb. 28, 2018	First version of `AUNTY`
Mar. 1, 2018	Mar. 15, 2018	Preparation of submission to ICIIA 2018
Apr. 15, 2018	May 28, 2018	Final version of `AUNTY`
May 3, 2018	May 28, 2018	Preparation of this manuscript

Chapter 2

Formalism and model of the heart

In this chapter we introduce our formalism by defining a syntax and an operational semantics. In order to show its usefulness we tackle a case study where we model part of the behavior of the heart. Specifically, we try and detect anomalies, associated with arrythmia, when analyzing data concerning BPM and duration of RR intervals. The contents of this chapter are based on our ACIIDS 2018 publication Calvo et al. (2018a).

The rest of this chapter is structured as follows. In Section 2.1 we introduce some preliminary notions concerning fuzzy relations and constraints. Section 2.2 presents the syntax of our formalism and its operational semantics. Section 2.3 presents our case study. We give our model of the behavior of the heart concerning BPM and duration of RR intervals, we review the methods used to prepare the dataset and to compute the values given by the model and provide some implementation details.

2.1 Preliminaries

The first concept that we need to present is *fuzzy relations*. They are similar to *boolean relations* but instead of returning true or false, they return a real value in the interval $[0, 1]$. The idea is that if we are sure that something holds then we have confidence equal to 1; otherwise, we will have a confidence less than 1, in particular, if we are sure that the relation does not hold then we have confidence equal to 0. There are a number of *standard* fuzzy relations (the obvious adaptations of classical numerical relations such as equality, less than or equal to, etc). In our work we need a fuzzy relation to classify values in intervals. In other words, we use this fuzzy relation to classify values with respect to *normal* behavior. For example, consider that the *normal* number of BPM of a healthy person must be between 63 and 70. If we observe that a

person always shows measures in this interval, then we can assume a normal behavior. If another patient sometimes shows values out of the interval but *close* to it then we can assume that with a high likelihood the patient is ok. However, if the patient shows many values out of the interval and relatively far from it, then we might order further experiments to try and detect a potential illness.

We define the relation $\overline{\alpha \leq \cdot \leq \beta}^{\delta}$ such that

$$\overline{\alpha \leq x \leq \beta}^{\delta} = \begin{cases} 1 & \text{if} \quad \alpha < x < \beta \\ 0 & \text{if} \quad x \leq \alpha - \delta \vee x \geq \beta + \delta \\ 0 & \text{if} \quad \beta < \alpha \\ 1 + \dfrac{x - \alpha}{\delta} & \text{if} \quad \alpha - \delta < x \leq \alpha \\ 1 - \dfrac{x - \beta}{\delta} & \text{if} \quad \beta \leq x \leq \beta + \delta \end{cases}$$

Intuitively, if a value x is such that $\alpha \leq x \leq \beta$ then we claim that the relation holds with confidence 1. If this is not the case and the distance from x to α or β is less than δ then we have a positive confidence (the confidence diminishes when the distance increases). Finally, if $x \notin [\alpha, \beta]$ and it is *far* from the interval then we have confidence 0 on x belonging to the interval. A simple example is $\overline{60 \leq x \leq 69}^{13}$. The idea is that if a patient is in the expected age range, that is $[60, 69]$, then the confidence on the obtained results is equal to 1. Otherwise, if the distance to the interval is more than 13 then the confidence is equal to zero. Finally, if the age is *close* to the interval, then the confidence linearly increases when the distance is reduced.

We combine confidence values by using *t-norms*.

Definition 1 *Let $T : [0,1] \times [0,1] \longrightarrow [0,1]$ be a function such that it satisfies:*

- *Commutativity: $T(a,b) = T(b,a)$.*
- *Associativity: $T(a, T(b,c)) = T(T(a,b), c)$.*
- *Identity element: $T(1,a) = T(a,1) = a$.*
- *Monotonicity: $T(a,b) \leq T(c,d)$ when $a \leq c$ and $b \leq d$.*

Then we say that T is a t-norm or triangular norm.

First, note that the definition implies $T(a,0) = 0$ for all a, since we have $T(a,0) \leq T(1,0) = 0$. Associativity allows us to easily consider n-ary versions of t-norms. Monotonicity will allow us to efficiently compute the maximum combined confidence. Next we give some examples of t-norms that can be found in the literature.

1. Gödel t-norm: $T(\delta_1, \delta_2) = min(\delta_1, \delta_2)$. We represent this t-norm with the symbol $\overline{\wedge}$.

2. Hamacher product t-norm: $T(\delta_1, \delta_2) = \dfrac{\delta_1 \cdot \delta_2}{\delta_1 + \delta_2 - \delta_1 \cdot \delta_2}$. We represent this t-norm with the symbol $\divideontimes$.

3. Łukasiewicz t-norm: $T(\delta_1, \delta_2) = \max(0, \delta_1 + \delta_2 - 1)$. We represent this t-norm with the symbol $\curlywedge$.

4. Product t-norm: $T(\delta_1, \delta_2) = \delta_1 \cdot \delta_2$. We represent this t-norm with the symbol $\bigstar$.

2.2 Fuzzy automata formalism

This section represents the bulk of the chapter. We present our formalism and its operational semantics. First, we define some notions that will be used in the definition of our automata.

In order to track some relevant data during the traversal of an automaton, we need notation to deal with *variables* and *variable transformations*.

Definition 2 *Let X be a set of variables taking values in $\mathbb{R}$. We define the set of variable transformations $\mathcal{VT}$ as the set of expressions assigning a value to each variable of the set. We will use the following notation*

$$[y_1/x_1, \ldots, y_m/x_m]$$

where each y_i is a real valued expression over the set of variables X and each x_i is a variable in X. The semantics of this transformation is that each x_i takes the value obtained after evaluating y_i (possible taking into account the current values of the variables in X); if a variable x_i does not appear in the expression then we have that the variable does not change its value after the transformation.

Transitions in our automata will be parameterized by a *fuzzy constraint*. Intuitively, a transition can be fired only if the grade of confidence of the constraint with respect to the provided parameter is greater than 0. The bigger this grade of confidence is, the higher our *confidence* in the results obtained by following this transition is.

Definition 3 *A* fuzzy constraint *is a formula where fuzzy relations are used instead of boolean relations and t-norms are used to combine relations instead of boolean operators. We denote by $\mathcal{FC}$ the set of fuzzy constraints. Let C be a fuzzy constraint with n parameters and $\overline{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$. We have that $\mu_C(\overline{x})$ denotes the* satisfaction degree *or* grade of confidence *(GoC) of C for $\overline{x}$.*

The main component of the transitions of our automata is the action labelling them. We distinguish between inputs and outputs. In general, *inputs* will be used to receive information about the state of the environment. We will use *outputs* to send messages back to the environment. For example, we can issue an alarm indicating that a potential problem has been found at a certain minute and with a certain GoC.

Definition 4 *Let* Acts *be a finite set of actions (they will be used to model the actions that a system can perform). We will distinguish between* inputs*, preceded by* ?*, and* outputs*, preceded by* !*.*

Finally, we introduce notation to define the set of all tuples of real numbers.

Definition 5 *Let* $\mathfrak{R}$ *be equal to* $\bigcup_{i\geq 1} \mathbb{R}^i$*, that is,* $\mathfrak{R}$ *is a set containing all the tuples, of any arity, with real number values.*

Next we introduce our notion of fuzzy automata, a revised version of previous work Camacho et al. (2017). Next we briefly discuss the main improvements with respect to the original formulation. We have included a tuple of variables as a parameter of the actions. This fact strongly simplifies, while keeping the same expressive power, the previous framework based on *fields*. We have simplified the operational semantics, removing a clause that gave priority to inputs over outputs. This was found to be both unnecessary and potentially problematic. We have clarified the way in which we obtain and process the data that we feed to the automaton. Moreover, the information returned for each patient after processing their data is structured in a more useful way. Specifically, we have disaggregated the obtained data and we currently provide different alternatives (together with its associated GoC).

Definition 6 *A* fuzzy automaton *is a tuple* $(S, \text{Acts}, X, s_0, T)$ *where:*

- S *is a finite set of states.*
- Acts *is a finite set of actions, partitioned into a set of inputs* I *and a set of outputs* O*.*
- X *is a set of variables ranging over* $\mathbb{R}$*. The set includes a variable* GoC*, which will be used to store the Hamacher grade of confidence associated with sequences of transitions. We assume that the initial value of* GoC *is* 1*.*
- s_0 *is the initial state.*
- $T \subseteq S \times (I \times X \cup O \times \mathfrak{R}) \times \mathcal{FC} \times \mathcal{VT} \times S$ *is the set of transitions. We assume that each transition implicitly applies the following variable transformation* $[\mu_C \circledast GoC/GoC]$*.*

Fuzzy automata are directed graphs where transitions have an associated condition, indicating the grade of confidence with which we can execute the transition, and a transformation of the variables. In addition, transitions can be labeled either by an input or by an output. Intuitively, a transition $(s, (a, \alpha), C, V, s') \in T$ denotes that if the automaton is in state s and receives/sends from/to the environment $a(\alpha)$, where a is an input/output action and α is a variable/tuple of positive real values, then the previous transition can be triggered if $\mu_C(\alpha) > 0$, the new values of the variables will be given by V, and the automaton will move to state s' . Usually, transitions labeled with an output will have a trivial fuzzy constraint (that is, it will be `True`).

Next, we are going to define the operational behavior of fuzzy automata. This *operational semantics* will be used to obtain their (fuzzy) traces. We start in the initial state of the automaton, produce actions and trigger a transition labelled by the action if the attached value is included in the fuzzy relation induced by the constraint. We decorate transitions with a real number $\epsilon \in [0, 1]$ indicating its certainty. First, we define a single transition and then we concatenate transitions to conform traces.

Definition 7 *Let $A = (S, \text{Acts}, X, s_0, T)$ be a fuzzy automaton and $\triangle$ be a t-norm. Given states $s_1, s_2 \in S$, we have a* transition *from s_1 to s_2, after performing the action $a \in \text{Acts}$ for α with confidence ϵ, denoted by $s_1 \xrightarrow{(a(\alpha),V)}_{\epsilon} s_2$, if the following conditions hold:*

- *There exists $C \in \mathcal{FC}$ such that $(s_1, (a, \alpha), C, V, s_2) \in T$.*
- *$\mu_C(\alpha) = \epsilon$ and $\epsilon > 0$.*
- *The new values of the variables belonging to X are given by $V \in \mathcal{VT}$.*

We say that a sequence $s_0 \xrightarrow{(a_1(\alpha_1),V_1)}_{\epsilon_1} s_1 \xrightarrow{(a_2(\alpha_2),V_2)}_{\epsilon_2} \cdots \xrightarrow{(a_n(\alpha_n),V_n)}_{\epsilon_n} s_n$ of consecutive transitions starting in the initial state of the automaton A is a $\triangle$-trace *of A if $\epsilon = \triangle\{\epsilon_1, \ldots, \epsilon_n\}$ is greater than zero and the values of the variables of X are the result of sequentially applying the variable transformations $V_1, \ldots, V_n$ to X. We call this composed variable transformation V. In this case we write $s_0 \xrightarrow{(a_1,\ldots,a_n,\alpha_1,\ldots,\alpha_n,V)}_{\epsilon} s_n$.*

Intuitively, a transition $(s, (a, \alpha), C, V, s')$ denotes that if the automaton is in state s and receives/sends from/to the environment $a(\alpha)$, where a is an input/output action and α is a variable/tuple of positive real values, then the previous transition can be triggered if $\mu_C(\alpha) = \epsilon > 0$, the new values of the variables will be given by V, and the automaton will move to state s'.

2.3 Case Study

In this section we present the application of our fuzzy automata in a real scenario: prediction of heart problems. We define the automaton *Heart*, which is able to alert about the level of risk of a patient. In order to produce a diagnosis, we use the available information and physical evidence collected from ECGs. The information managed by the automaton is:

- Gender. We have 2 groups: Men and Women.
- Age. We have 8 groups of age.
- Heartbeats. The range of correct beats per minute (BPM) for healthy patients, according to their gender and age.
- RR intervals. The range of correct RR interval duration (measured in milliseconds) for healthy patients, according to their gender, age and BPM.

Additionally, we consider that our set of actions consists of the following operations:

- $?checkGender(gen)$. It reads the gender of the patient.
- $?checkAge(age)$. It reads the age of the patient.
- $?minute(m)$. It reads the current minute of the recording.
- $?readBPM(bpm)$. It reads the amount of beats in the current minute.
- $?readRR(rr)$. It reads the next RR interval in the current minute.
- $?noMorePendingRR(\cdot)$. It receives a notification that there are no more RR intervals in the current minute.
- $?endOfRecord(\cdot)$. It receives a notification to denote that there are no more minutes in the analyzed record.
- $!recordAgeRange(x, y)$. It indicates that the data will be analyzed in the age group of patients between x and y years. Note that a patient can be analyzed, with different grades of confidences on the obtained results, in different age groups.
- $!recordAlarm(min, GoC)$. It indicates, with a GoC equal to GoC, that an alarm will be raised in the current minute.
- $!ok(min, GoC)$. It indicates, with a GoC equal to GoC, that no alarm will be recorded for the current minute.

First, the automaton receives the gender and age of the patient. Next, for each minute, the environment sends a sequence of values and the automaton produces a diagnosis (*ok* or *alarm*). These sequences are formed by the actual number of *beats* recorded in the minute (BPM, one value per minute) followed by the length of each *RR interval*. Therefore, for each patient we obtain a sequence of n ok/alarm messages, being n the number of minutes in the record, labeled with the associated minute and the grade of confidence on the validity of the result.

Example 1 *Consider the component of the automata* Heart *given in Chapter 3 Figure 3.4 where we assume that the value* 0 *denotes males and* 1 *denotes females. For example, we could observe a trace such as*

$$(?checkGender(0)), (?checkAge(65)), (!recordAgeRange(60, 69)),$$
$$(?minute(1)), (?readBPM(62)), (?readRR(977)), (?readRR(968)),$$
$$(\cdots), (?noMorePendingRR()), (!ok(1, 1.0))$$

as the result of having the automaton working during a minute by analyzing a sample of a 65 years old male patient. As usual, inputs are preceded by ?*, outputs are preceded by* ! *and* $(\cdots)$ *indicates that some* $?readRR$ *actions have been omitted from the trace due to presentation purposes.*

Initially, our automaton has two transitions: one per gender. After that, each branch has 8 transitions, one per age group. Each of these age groups has an associated sub-automaton with 8 states and a transition to a common *final* state (the state q_{115} in Figure 3.4) to denote that all the data for this patient have been processed. Each minute, the automaton checks if the number of beats falls within the normal amount of beats per minute in the age range. If it does, then the automaton does not take into account the RR intervals in that minute and signals that minute to be *ok*. Otherwise, the automaton processes each RR interval. If there is at least one interval out of range, an alarm is raised (with a certain grade of confidence). As previously commented, we use variables to track some data. More specifically, in addition to the *built-in* variable *GoC*, we use a variable called *branchGoC*. In every transition entering an *age branch*, the recently computed GoC value is saved in the *branchGoC* variable. At the end of each minute, that value is stored in the *GoC* variable. Therefore, the GoC values associated with each minute are not affected by the values corresponding to the previous minutes. The value received when performing a *?minute* input action is also stored, in the variable *min*, and returned after processing the corresponding minute data. If at any moment of the study the state q_{41} is reached, then the automaton will record the current state of the patient as a case in which he suffers the risk of having a heart problem. For each minute, we process data until we find a potential problem during that minute slot (that is, the state q_{41} is traversed), having the samples a duration of around 30 minutes.

Therefore, for each patient, the number of alarms that the automaton can raise is bounded by the number of minutes in the recording.

The values used to define our fuzzy constraints are taken from previous work. Normal ranges for heartbeats per minute, classified by gender, have been gathered from the work of Rijnbeek et al. Rijnbeek et al. (2014). In the case of the age, the δ value is obtained from the 20% of the highest value of each age range. The idea is that it is possible to wrongly classify a patient according to their age. For example, if we have a 53 years old male patient then we should classify him in the age group between 50 and 59. In addition, it may happen that the patient has a very healthy life style and, therefore, their heart is *younger*. Therefore, we should also classify them in the previous age group and decide whether their recorded data fits better in their *real* age group or in the *closer one* to the real one.

In the case of heartbeats, we had the median, 2nd percentile and 98th percentile from the database, but they were not applicable as limits for our automaton because they are not characteristic data of the sample of patients. For that reason, we applied the estimations made by Hozo et al. Hozo et al. (2005): if the size of a sample exceeds 25 then the median itself is the best estimator for the mean and the best estimator for the standard deviation is

$$\sigma \approx \frac{b-a}{6}$$

where a is the smallest value of the sample (the 2nd percentile in our case) and b is the largest value of the sample (the 98th percentile in our case). Fortunately, the database that we use Rijnbeek et al. (2014) has information from 13354 patients and, therefore, we can base our limits on the median of each range while δ is based on the standard deviations of each range.

Concerning RR intervals, we have used the data from the work of Haarmark et al. Haarmark et al. (2010). The problem in this case was that we only had the information of the RR intervals duration for the patients of the age range $[30-39]$. So, if we had used these limits for all the patients, then the prediction would have been erroneous. Therefore, we also considered another related work Khachaturian et al. (1972) where the duration of the RR intervals is derived from data corresponding to heartbeats. So, our limits are based on the application of the following formula

$$RR_{ms} \approx \frac{60000}{bpm}$$

to the BPM data obtained from the work of Rijnbeek et al. Rijnbeek et al. (2014).

2.3.1 Methodology and results

Next, we review the main implementation details behind our case study. In order to obtain the data that we feed to our automaton, we used the

`WFDB Software` Moody (2016) to extract *Inter-beat (RR) intervals* from the dataset that we consider in our case study Moody y Mark (2001). We included calls to the functionalities `sqrs` and `ann2rr` in our patient data loading script. We obtain several csv (comma-separated values) files for each patient's header and record files. These two files are later used by our trace generating script, which produces the data sent to the automaton. Essentially, we format the data in a way that can be easily processed by our automaton.

In order to assess the usefulness of our automaton, we used the MIT/BIH Arrhythmia Database Directory Moody y Mark (2001) `https://physionet.org/physiobank/database/mitdb/`. This study includes 48 ECGs recordings with a duration of 30 minutes from the Massachusetts Institute of Technology - Beth Israel Hospital arrhythmia database. All of them present some heart pathology. Specifically, 48% of the samples have been annotated in the database as representative cases of routine clinical recordings while the remaining 52% reflect uncommon cases of arrhythmias. As an example of the obtained results, in Appendix B we show minute data from each applicable age branch of the patients 100 and 104. Each cell contains two numbers: the first one is the Hamacher GoC of sending an *ok* signal while the second one is the Hamacher GoC of raising an *alarm*, both referring to that minute and age/gender branch. This table clearly shows why our new approach represents a big step forward with respect to previous work of the research group concerning the analysis of heart data Camacho et al. (2017): we are able to produce and appropriately process several dozens of *ok/alarm* signals.

Let us remark that these two patients are interesting because they show characteristics of an age group different to the one that their age shows. Specifically, patient 100 is male, between 60 and 69 years, but we obtain relevant grades of confidence for values in the age range of the 70s. This might imply that the patient is suffering premature aging of the heart tissue because his ECGs represent patterns of older people. On the contrary, patient 104 is a female between 60 and 69 years with high grades of confidence correspond to a woman on her 50s. This might indicate that the patient is a sport female that, regardless of having a healthy life style, suffers some heart pathologies corresponding to a younger person. In both cases, the value of the corresponding *?checkAge* constraint is positive for four branches: $(50 \leq age \leq 59)$, $(60 \leq age \leq 69)$, $(70 \leq age \leq 79)$ and $(age > 80)$. The table is formed by four columns for patient 100 and four columns for patient 104. Each row contains the GoC obtained in a minute. The record from patient 100 is 30 minutes long while the record from patient 104 is 29 minutes long. Let us briefly comment on the results obtained for these two patients. First, we notice that in both cases the maximum confidence is obtained in the $60-69$ branch. This means that there are not many RR or BPM values close to the limit of the normal range. The only case in which there is more

confidence outside the $60-69$ age branch is in the $!ok$ results of the patient 100. We can see that the columns corresponding to the age ranges $70-79$ and >80 present higher confidence in these cases. These values can indicate that the values observed from the heart of patient 100 could be normal for a much older person. This idea is reinforced by the observation that in the >80 branch the confidence in the $!ok$ case is higher that the confidence in the $!alarm$ case. The most relevant difference we observe between these two patients is that patient 100 is outside his normal parameters in every minute, while patient 104 is showing a normal behavior most of the time, having some eventual alerts.

2.3.2 GoC computations

In this section we describe some of the technical aspects behind the process of the data. First, we show the part of the csv (comma separated values) file that we used to populate the automaton:

```
percentil, gender, 10s, 20s, 30s, 40s, 50s, 60s, 70s, 80s
50, male, 73, 65, 65, 66, 67, 67, 67, 74
2, male, 49, 45, 46, 47, 48, 48, 50, 40
98, male, 107, 94, 95, 95, 94, 95, 99, 97
50, female, 72, 67, 66, 67, 69, 71, 72, 72
2, female, 47, 48, 47, 47, 52, 53, 55, 50
98, female, 105, 98, 95, 90, 94, 94, 98, 102
```

This csv file was firstly processed using the python script given in Figure 2.1. Essentially, this script produces the ranges of values for healthy patients. A file named *patients.csv* was composed with the number of each patient, his/her age and his/her gender. The Python script given in Figures 2.2 and 2.3 generates the GoC for each patient in the file. We used two additional scripts, *fuzzyRestrictions.py* and *tnorms.py*, to compute the relations and the t-norms used in the model. They can be found in Figure 2.4.

```
f = open('standardlimits/bpm.csv','r')
extremos = []
medianas = []
f.readline()
#masculinos
medianas += f.readline().split(',')[2:]
aux1 = f.readline().split(',')[2:]
aux2 = f.readline().split(',')[2:]
extremos += zip(aux1,aux2)
#femeninos
medianas += f.readline().split(',')[2:]
aux1 = f.readline().split(',')[2:]
aux2 = f.readline().split(',')[2:]
extremos += zip(aux1,aux2)
f.close()
medianas = [float(i) for i in medianas]
extremos = [(float(i),float(j)) for i,j in extremos]
desviaciones = [(b-a)/6 for a,b in extremos]
deltasBPM = [i/2 for i in desviaciones]
supsBPM = [i+d for i,d in zip(medianas,deltasBPM)]
infsBPM = [i-d for i,d in zip(medianas,deltasBPM)]
#aplicamos aqui datos de [20]
deltasRR = [84.5 for i in range(8)] + \
    [74. for i in range(8)]
infsRR = [60000/i - d
 for i,d in zip(medianas,deltasRR)]
supsRR = [60000/i + d
 for i,d in zip(medianas,deltasRR)]
```

Figure 2.1: Python script to generate intervals

```
import fuzzyRestrictions as fr
import confidenceLimits as cl
import tnorms as tn

pat = open("mitdb/patients.csv", "r")

for lin in pat.readlines():
 tr = open("traces/" + lin.split(',')[0] + \
    "confidence.csv", "w")
 gend = int(lin.split(',')[2][:-1]) * 8

 for j in range(8):
  if j == 0:
   ageinterval = fr.fuzzyleq(19, 4)
  elif j == 7:
   ageinterval = fr.fuzzygeq(80, 16)
  else:
   ageinterval = fr.fuzzyinterval(19+j*10,
      10+j*10,0.2*(19+j*10))
  if ageinterval(int(lin.split(',')[1])) > 0:
   tr.write(str(19+10*j if j < 7 else 80)+",
  "+str(ageinterval(int(lin.split(',')[1])))+"\n")
  else:
   continue

  supBPM = cl.supsBPM[gend+j]
  infBPM = cl.infsBPM[gend+j]
  deltaBPM = cl.deltasBPM[gend+j]
  supRR = cl.supsRR[gend+j]
  infRR = cl.infsRR[gend+j]
  deltaRR = cl.deltasRR[gend+j]

  rrfile = open("mitdb/" + lin.split(',')[0] + \
      "rr.csv", "r")
```

Figure 2.2: Python script to process patients (1/2)

```
  time = 0
  for i in [i+1 for i in range(30)]:
   beats = 0
   RRs = []
   endoffile = 0
   while time < 60000*i:
    beats += 1
    aux = rrfile.readline().rstrip()
    if aux == "":
     endoffile = 1
     break
    RRs += [int(float(aux)*1000)]
    time += RRs[-1]
   if endoffile == 1:
    break
   tr.write("m"+str(i))
   tr.write("\n")
   tr.write(str(fr.fuzzyinterval(supBPM,
      infBPM,deltaBPM)(beats)))
   tr.write(",")
   tr.write(str(fr.fuzzyantiinterval(supBPM,
      infBPM,deltaBPM)(beats)))
   tr.write("\n")
   for rr in RRs[:-1]:
    tr.write(str(fr.fuzzyantiinterval(supRR,
      infRR,deltaRR)(rr)))
    tr.write(",")
   tr.write(str(fr.fuzzyantiinterval(supRR,
      infRR,deltaRR)(RRs[-1])))
   tr.write("\n")
   tr.write(str(tn.hamacher([fr.fuzzyantiinterval(supRR,
     infRR,deltaRR)(rr) for rr in RRs])))
   tr.write("\n")
  rrfile.close()
 tr.close()
pat.close()
```

Figure 2.3: Python script to process patients(2/2)

```
def fuzzyleq(y,delta):
 return lambda x: 1 if x <= y else \
   0 if x > y + delta else \
   1 - (x - y) / delta

def fuzzyor(lam1, lam2):
 return lambda x: max([lam1(x), lam2(x)])

def fuzzyand(lam1, lam2):
 return lambda x: min([lam1(x), lam2(x)])

def fuzzynot(lam1):
 return lambda x: 1 - lam1(x)

def fuzzygeq(z,delta):
 return fuzzynot(fuzzyleq(z-delta,delta))

def fuzzyinterval(y,z,delta):
 return fuzzyand(fuzzyleq(y,delta),
     fuzzygeq(z,delta))

def fuzzyantiinterval(y,z,delta):
 return fuzzyor(fuzzygeq(y,delta),
      fuzzyleq(z,delta))
```

```
def hamacher(arr):
 res = 1.
 for i in arr:
  if i == 0. or res == 0.:
   return 0.
  res = (res * i) / (res + i - res*i)
 return res
```

Figure 2.4: Python script to process fuzzy constraints and t-norms

Chapter 3

AUNTY: AUtomatically aNalyze daTa using fuzzY automata

In this chapter we introduce our AUNTY tool. The main purpose of our tool is to fully support the framework to specify and analyze complex systems where information has to be treated in a fuzzy way that we presented in Chapter 2 of this Thesis. We will shown its internal architecture and briefly described its GUI so that potential users can use this chapter as a *small tutorial*. Some of the material presented in this chapter has been taken from our paper Calvo et al. (2018b) published in the proceedings of ICIIA 2018.

The rest of the chapter is structured as follows. In Section 3.1 we describe how AUNTY was born and we give a high-level enumeration of its main features. In Section 3.2 we present the (software) architecture of AUNTY and review its features as included in the GUI of the tool. Section 3.3 gives the BNF (Backus-Naur form) definitions of the most relevant concepts used in AUNTY. Finally, in Section 3.4 we review the internal format used to represent automata in AUNTY.

3.1 Description of AUNTY

When the software that we implemented to perform the experiments in our ACIIDS 2018 paper Calvo et al. (2018a) became stable, we decided to develop a tool supporting the specification and analysis of systems described by using our version of fuzzy automata. Actually, although our case study was fully supported by a dedicated computer program, we thought that it would be much more useful to have a generic tool where all the current and future case studies could be performed under a common umbrella. The main features of the tool are:

- AUNTY allows users to graphically represent systems where fuzzy logic is used to take decisions.

- AUNTY provides a framework to automatically analyze whether a dataset shows some abnormal behavior.
- AUNTY can be used to specify a system that gives alternative explanations to a particular behavior.
- AUNTY is built in a modular and extensible way, facilitating the task of dealing with diverse datasets and models.

Regarding the last point, AUNTY processes data expressed in its internal formats, so the first layer between the tool and the data source must transform raw data into a sequence of actions that is recognizable by the automaton.

3.2 Design and main features of AUNTY

In this section we briefly describe the architecture of the AUNTY tool and present its main features by going through its graphical interface. AUNTY has being developed using the programming language Python. We have chosen this programming language because its wide use allows potential users of the tool to easily extend it with new formats to import data and with new t-norms. In addition, the interface of the tool has been developed using PyQt.[1]

3.2.1 Architecture of AUNTY

Our main concern during the development of AUNTY was that our tool could be widely usable. Therefore, it is quite important to have a proper separation of concerns between the main components that conform the tool (see Figure 3.1). Next, we will briefly discuss the main *responsibilities* held by each component of the tool.

- *Automata manager*. This component is in charge of managing the internal representation of the automaton used to process the data. It also tells the *trace applicator* the transitions that can be taken, if a given action can be performed, and with which confidence. This component uses Graphviz[2] to give a graphical representation of the automaton, transforming its internal representation to a graph expressed in the DOT language.
- *Trace generator*. This component is the only one that needs to know about the format of the data source. Its main function is to translate that data into a sequence of input actions (that is, a *trace*).

[1] https://riverbankcomputing.com/software/pyqt/intro
[2] http://www.graphviz.org/

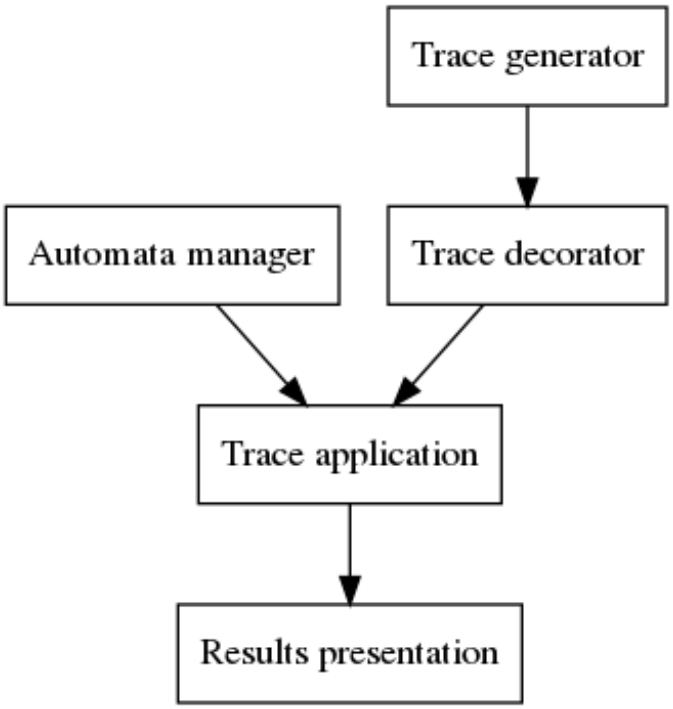

Figure 3.1: Architecture of AUNTY

- *Trace decorator.* The traces generated by the *trace generator* only have input actions. However, systems also have output actions that will be used, for example, to indicate that a certain result has been produced. The process of trace decoration allows the tool to consider only those traces that include a given output action.

- *Trace application.* This component manages the variable transformations and computes the highest possible grade of confidence that the given trace could reach after traversing the automaton.

- *Results presentation.* This component communicates the conclusions to the user by providing a measure of its certainty and the values obtained in the variables. Actually, this component will facilitate that the tool shows multiple alternatives in a tabular way.

3.2.2 Graphical user interface

Even though we would like that AUNTY is a versatile and general purpose tool, our main priority concerning usability when we designed our tool was that AUNTY were an intuitive and easy to use graphical user interface (see

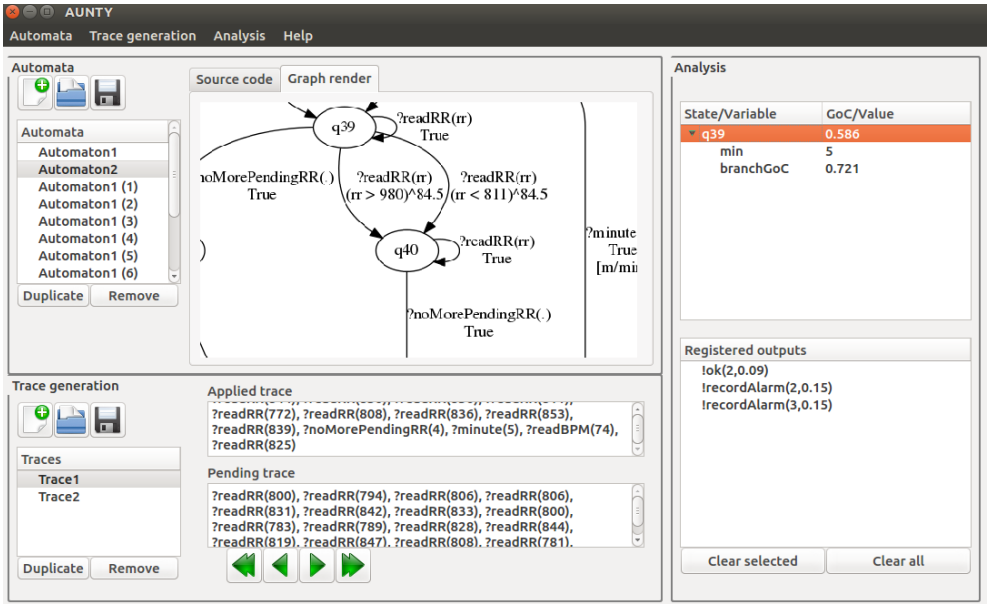

Figure 3.2: Graphical user interface

Figure 3.2 for a screenshot of our tool). The interface presents three distinguished areas:

- *Automata.* The automata area shows a list of loaded automata and renders their graphs by using PyGraphviz.[3] This area of the GUI shows the loaded automata in an editable text form and offers two dialogs to add or remove individual transitions. Automata can be loaded from a text file and saved after editing.

- *Trace generation.* The trace generation area shows the list of loaded traces corresponding to the selected automaton. Traces can be loaded from the dataset, selecting a patient in the dialog shown in Figure 3.3. Each trace can be applied to the corresponding automaton, recording results in the *analysis* area. Trace execution can be played step by step, fast forwarded and rewinded. The selected trace is shown in two separated text areas. The fragment of the trace that has been already applied to the automaton is shown in the upper one, which is not editable. The rest of the trace is shown below and can be modified before applying it. This second fragment of the trace may contain unspecified actions and parameters, producing different concrete traces when applied. Trace application can also be undone step by step, allowing the user to modify the last applied actions.

[3] `https://pygraphviz.github.io/`

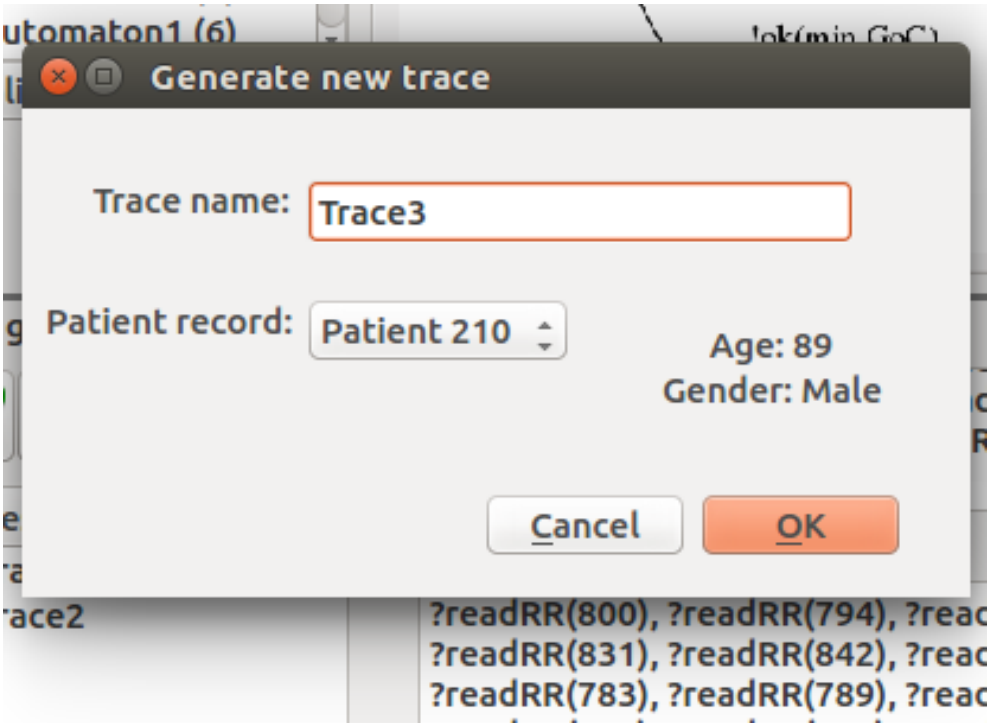

Figure 3.3: Trace generation dialog

- *Analysis.* This region shows every state in which the automaton can be after the current application of the trace. Each state shows both the confidence to reach it and the variables obtained after reaching it. Selecting a state changes the applied trace shown to the concrete one used to reach that state with maximum confidence. Below, a list of output actions produced during the application of the selected trace is shown. That list included these actions even if the trace is rewinded, showing a record of alternative executions.

3.3 Syntax of the languages in AUNTY

In this section we review the different BNF expressions used to represent some of the concepts of AUNTY. We start with the formal syntax to define terms and binary/ternary fuzzy relations (the definition of relations with another arity follows the same pattern).

Definition 8 *We denote by V the syntactic category of variable names. We denote by R the syntactic category of real valued constants. We denote by R_+ the syntactic category of non-negative real valued constants. A* term *is*

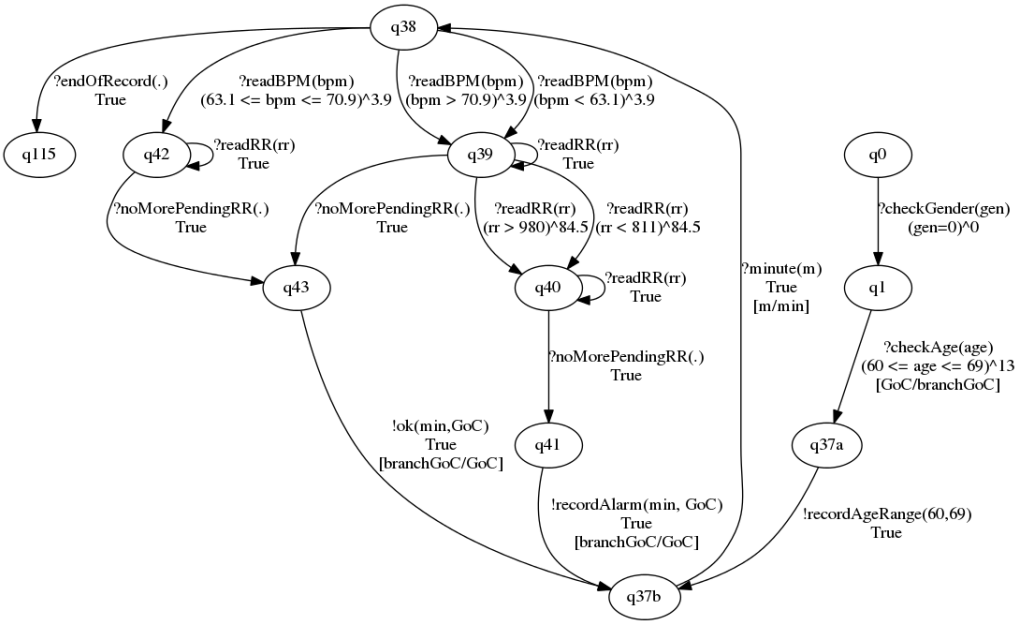

Figure 3.4: Sub-automaton used to analyze heart data (male patients between 60 and 69 years old)

any word built from the following BNF.

$$< Term > ::= < V > \ | \ < R >$$

We denote by $\mathcal{T}$ *the set of all terms.*

A binary fuzzy relation *is any word built from the following BNF.*

$$\begin{aligned} < B > ::= & \text{“(”} < Term > \text{“<=”} < Term > \text{“)^”} < R_+ > \ | \\ & \text{“(”} < Term > \text{“>=”} < Term > \text{“)^”} < R_+ > \ | \\ & \text{“(”} < Term > \text{“==”} < Term > \text{“)^”} < R_+ > \end{aligned}$$

A ternary fuzzy relation *is any word built from the following BNF.*

$$< T > ::= \text{“(”} < Term > \text{“<=”} < Term > \text{“<=”} < Term > \text{“)^”} < R_+ >$$

We denote by $\mathcal{R}$ *the set of all binary and ternary fuzzy relations.*

We also need to introduce the following notation related to terms, variables and relations.

Definition 9 *Let* $b \in \mathcal{R}$ *and* $t \in \mathcal{T}$*. We denote by* $FV(b)$ *and* $FV(t)$ *the set of terms from* V *appearing in* b *and* t*, respectively.*

Let $v \in V$ *and* $c \in R$*. We denote by* $b[c/v]$ *and* $t[c/v]$ *the fuzzy relation resulting from the substitution of every occurrence of* v *in* b *and* t *by*

```
q38, ?endOfRecord(), True, [], q115;
q42, ?noMorePendingRR(), True, [], q43;
q42, ?readRR(rr), True, [], q42;
q38, ?readBPM(bpm), (63.1 <= bpm <= 70.9)^3.9, [], q42;
q0, ?checkGender(gen), (gen == 0)^0, [], q1;
q37a, !recordAgeRange60to69(), True, [], q37b;
q1, ?checkAge(age),(60<=age<= 69)^13,[GoC/branchGoC],q37a;
q37b, ?minute(m), True, [m/min], q38;
q38, ?readBPM(bpm), (bpm >= 70.9)^3.9, [], q39;
q38, ?readBPM(bpm), (bpm <= 63.1)^3.9, [], q39;
q39, ?readRR(rr), (rr >= 980)^84.5, [], q40;
q39, ?readRR(rr), (rr <= 811)^84.5, [], q40;
q39, ?readRR(rr), True, [], q39;
q43, !ok(min,GoC), True, [branchGoC/GoC], q37b;
q41, !recordAlarm(min,GoC), True, [branchGoC/GoC], q37b;
q40, ?noMorePendingRR(), True, [], q41;
q40, ?readRR(rr), True, [], q40;
q39, ?noMorePendingRR(), True, [], q43;
```

Figure 3.5: Code corresponding to the sub-automaton given in Figure 3.4

the constant c, respectively. We use $b[c_1/v_1, \ldots, c_n/v_n]$ *as a shorthand for* $b[c_1/v_1] \ldots [c_n/v_n]$.

We consider the four t-norms defined in Section 2.1 but more t-norms can be easily added.

Definition 10 *The syntactic category of t-norms is given by the following BNF.*

$$< Tnorm > ::= \text{“}Luka\text{”} \mid \text{“}Gode\text{”} \mid \text{“}Prod\text{”} \mid \text{“}Hama\text{”}$$

Finally, the syntax to define fuzzy constraints is given in the following definition.

Definition 11 *A fuzzy constraint is any word built from the following BNF.*

$$< C > ::= \text{“}True\text{”} \mid < C >< Tnorm >< C > \mid < B > \mid < T >$$

Over this syntactic category, and in the standard way, we compositionally define the set of free variables, the grade of conficence degree and variable substitution. Note that since two different t-norms may not be mutually associative, the concrete syntax of constraints may need to use parenthesis to disambiguate the order of application of t-norms.

```
grammar = """
Automaton : tr*= Transicion ;
Transicion: s1=ID ',' act=Action args=Varlist ','
            constr=C ',' vtran=VT ',' s2=ID ';' ;

Term : V | R;
V    : name=ID;
R    : value=FLOAT;
B    : Bleq | Bgeq | Beq;
Bleq : '(' t1=Term '<=' t2=Term ')^' delta=FLOAT ;
Bgeq : '(' t1=Term '>=' t2=Term ')^' delta=FLOAT ;
Beq  : '(' t1=Term '==' t2=Term ')^' delta=FLOAT ;
T : '(' t1=Term '<=' t2=Term '<=' t3=Term ')^' delta=FLOAT;
Tnorm: 'Luka' | 'Gode' | 'Prod' | 'Hama';
C    : 'True' | c1=C tnorm=Tnorm c2=C | B | T;

VT   : '[' head=Trans tail*=TransTail ']' | '[]';
TransTail : ',' head=Trans;
Trans     : c=Term '/' v=V;

Varlist   : '(' head=V tail*=VarTail ')';
VarTail   : ',' head=V;

Constlist : '(' head=R tail*=VarTail ')';
ConstTail : ',' head=R;

Action    : '?' ID | '!' ID;

"""
```

Figure 3.6: Code used to generate the grammar

3.4 Automaton definition language

In order to construct a new model, the users may define its automaton by providing a textual representation of its transitions. The user does not need to explicitly enumerate the sets of states and actions because they are automatically extracted from the set of transitions. As an example of the aspect of this representation, in Figure 3.5 we show the code that defines the subautomaton of the Heart automaton that appears in Figure 3.4.

AUNTY uses *textX* Dejanović et al. (2017) to parse this language. This Python library is called to construct a metamodel from a given grammar, which is used to construct syntax trees, represented as a hierarchy of Python objects, from strings accepted by the grammar. The code given in Figure 3.6 shows how the language is defined.

One of the nice features of AUNTY is its ability to show a graphical repre-

```
digraph automata {
"q38" -> "q115" [label="?endOfRecord()\nTrue"]
"q42" -> "q43" [label="?noMorePendingRR()\nTrue"]
"q42" -> "q42" [label="?readRR(rr)\nTrue"]
"q38" -> "q42" [label="?readBPM(bpm)\n(63.1<=bpm<=70.9)^3.9"]
"q0" -> "q1" [label="?checkGender(gen)\n(gen == 0.0)^0.0"]
"q37a" -> "q37b" [label="!recordAgeRange60to69()\nTrue"]
"q1" -> "q37a"
[label="?checkAge(age)\n(60<=age<=69)^13\n[GoC/branchGoC]"]
"q37b" -> "q38" [label="?minute(m)\nTrue\n[m/min]"]
"q38" -> "q39" [label="?readBPM(bpm)\n(bpm >= 70.9)^3.9"]
"q38" -> "q39" [label="?readBPM(bpm)\n(bpm <= 63.1)^3.9"]
"q39" -> "q40" [label="?readRR(rr)\n(rr >= 980.0)^84.5"]
"q39" -> "q40" [label="?readRR(rr)\n(rr <= 811.0)^84.5"]
"q39" -> "q39" [label="?readRR(rr)\nTrue"]
"q43" -> "q37b"
[label="!ok(min,GoC)\nTrue\n[branchGoC/GoC]"]
"q41" -> "q37b"
[label="!recordAlarm(min,GoC)\nTrue\n[branchGoC/GoC]"]
"q40" -> "q41" [label="?noMorePendingRR()\nTrue"]
"q40" -> "q40" [label="?readRR(rr)\nTrue"]
"q39" -> "q43" [label="?noMorePendingRR()\nTrue"]
}
```

Figure 3.7: DOT code automatically generated corresponding to Figure 3.4

sentation of our fuzzy automata. This is done by transforming the internal representation of the automaton into the DOT language, which is used to plot graphs using Graphviz (these plots look similar to the one shown in Figure 3.4). For example, the DOT code given in Figure 3.4 is automatically generated from the definition of the subautomaton corresponding to male patients in the age range between 60 and 69 years. The transformation from our automaton definition language into the DOT language is performed using the functions given in Appendix A.

Chapter 4

Conclusions

In this Thesis we have introduced a variant of finite automata where constraints, indicating whether a certain transition can be performed, are evaluated under a *fuzzy* point of view. We have implemented a model of the heart that takes into account data about the beats per minute and the duration of RR intervals. In order to decide whether potential dangers have been observed, we used information about the gender and age of the patients. In the latter case, we have also used a *fuzzy* approach because a patient can be classified in several age groups. We have also introduced `AUNTY`: a tool to fully support the formal framework. We have shown its internal architecture and briefly described its GUI. The results of this Thesis have been published in two international conferences Calvo et al. (2018a,b).

The rest of the chapter is structured as follows. In Section 4.1 we discuss the contributions of the Thesis while in Section 4.2 we briefly describe some lines for future work.

4.1 Review of the contributions

We set three main goals when we initiated the work reported in this Thesis.

1. Define a *fuzzy* extension of the classical finite automata formalism.
2. Show the usefulness of the formalism by using it in the analysis of a non-trivial system.
3. Provide a tool supporting the use of the formalism.

All the goals have been achieved but the contribution of this work to the state-of-the-art is not the same for all of them. First, we have a new formalism. The impact of the new formalism is not big if we simply consider the definition of its syntax and its operational semantics, although it is worth to mention that the simplification of the operational semantics facilitates

the definition of more difficult semantic frameworks, as discussed in the next section. Once said this, the impact of the achievement of this goal is still important if we consider that the information provided in the new framework after *feeding* a sequence of actions to an automaton is more useful than the one obtained in the framework that we took as starting point Camacho et al. (2017).

The last mentioned contribution strongly influenced the impact of the second goal. In fact, our case study confirmed that we give more useful information than the previous work Camacho et al. (2017): we return different degrees of confidence, in an organized manner, so that the conclusions about the analyzed data are more relevant.

Finally, the achievement of the third goal is a very important contribution because it provides a tool to specify and analyze systems where fuzzy information must be taken into account. One of the main critics about formal methods is that theory and practice do not usually go together. Although the main problem is probably due to the difficulty of understanding formal methods without the appropriate mathematical background, it is also true that the lack of good tools is a detriment to its wide use in industry.[1] Therefore, providing a tool, although with the obvious limitations of being a tool developed by an undergraduate student, supporting our formal framework is one of the most relevant and interesting contributions of our work.

4.2 Future work

There is room to work on extensions of the two main lines of work reported in this Thesis. Next, we briefly enumerate the lines that we have already identified.

Currently, the formal framework is useful to specify systems and analyze data but there is a line of work that we would like to continue in the near future. Specifically, we would like to study a formal framework to test from our fuzzy automata. The idea is to define a conformance relation to decide whether a system under test correctly implements a specification. We will take as starting point the work developed for the previous version of our fuzzy automata Boubeta-Puig et al. (2017b).

Concerning `AUNTY`, this Thesis does not represent the end of its development. In fact, we already contemplate several lines for future work. First, the tool can be expanded to provide other features. For example, after adding some capabilities, `AUNTY` may be transformed into a *recommender system*. We still have to evaluate the usefulness of the tool with different case studies so that we can find hidden *weaknesses* and fix them. We also would like to obtain more data from patients with the aim of applying techniques, such

[1] There are some interesting examples of the successful use of formal methods in industry Boulanger (2013); Newcombe et al. (2015).

as evolutive algorithms, swarm intelligence and neural networks, to improve the ability of our automata to detect illnesses. This last goal should be pursued in cooperation with researchers in Medicine, working on alternative models where the *classification* of patients considers characteristics such as size/weight and medical record.

Appendix A

Functions to transform into DOT code

```
def Tstr(t):
    if tname(t) == 'R':
        return str(t.value)
    else:
        return t.name

def vt2list(vt):
    if vt.head == None:
        return []
    else:
        return [vt.head]+[a.head for a in vt.tail]

def vt2string(vt):
    return "["+str.join(",",[Tstr(t.c)+"/"+ \
           Tstr(t.v) for t in vt2list(vt)])+"]"

def C2string(C):
    if C == 'True':
        return C
    if tname(C) == 'Bleq':
        return  '('+ Tstr(C.t1)+ ' <= ' +Tstr(C.t2) + \
                ')^' + str(C.delta)
    if tname(C) == 'Beq':
        return  '('+ Tstr(C.t1)+ ' == ' +Tstr(C.t2) + \
                ')^' + str(C.delta)
    if tname(C) == 'Bgeq':
        return  '('+ Tstr(C.t1) +' >= ' +Tstr(C.t2) + \
              ')^' + str(C.delta)
    if tname(C) == 'T':
        return  '('+ Tstr(C.t1)+ ' <= ' +Tstr(C.t2) + \
              ' <= ' + Tstr(C.t3) + ')^' + str(C.delta)
```

```
    if tname(C) == 'Compo':
        return "(" + C2string(C.c1) + " " + \
           C.tnorm + " " + C2string(C.c2) + ")"

def varlist2list(varlist):
    if varlist.head == None:
        return []
    else:
        return [varlist.head.name]+ \
            [a.head.name for a in varlist.tail]

def varlist2string(varlist):
    return "("+str.join(",",varlist2list(varlist))+")"

def automata2dot(autom):
    res = "digraph automata {"
    for tr in autom.tr:
        res = res + '"'+tr.s1+'" -> "'+tr.s2 + \
        '" [label="'+tr.act+varlist2string(tr.args)+ \
        '\\n'+C2string(tr.constr) + \
        ('' if tr.vtran.head == None else
         '\\n' + (vt2string(tr.vtran)) )+ \
        '"]\n'
    return res + "}"
```

Appendix B

Results corresponding to patients #100 and #104

In the following table we show the results corresponding to patients #100 and #104. Each cell includes a pair *GoC ok*/*GoC alarm* computed from the data observed during that minute.

	#100				#104			
	(50-59)	**(60-69)**	**(70-79)**	**(>80)**	**(50-59)**	**(60-69)**	**(70-79)**	**(>80)**
min. 1	0.09/0.15	0.21/1.00	0.28/0.94	0.31/0.00	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 2	0.09/0.15	0.21/1.00	0.28/0.94	0.31/0.00	0.00/0.41	0.54/1.00	0.68/0.75	0.12/0.12
min. 3	0.00/0.15	0.00/1.00	0.00/0.94	0.31/0.22	0.00/0.41	0.00/1.00	0.00/0.75	0.00/0.12
min. 4	0.09/0.15	0.21/1.00	0.28/0.94	0.31/0.00	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 5	0.09/0.15	0.21/1.00	0.28/0.94	0.31/0.00	0.00/0.41	0.00/1.00	0.29/0.75	0.12/0.12
min. 6	0.00/0.15	0.00/1.00	0.04/0.94	0.31/0.14	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 7	0.00/0.15	0.00/1.00	0.00/0.94	0.28/0.31	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 8	0.00/0.15	0.00/1.00	0.00/0.94	0.28/0.31	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 9	0.00/0.15	0.00/1.00	0.00/0.94	0.31/0.22	0.20/0.41	0.83/1.00	0.75/0.65	0.12/0.12
min. 10	0.00/0.15	0.00/1.00	0.00/0.94	0.31/0.26	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 11	0.00/0.15	0.00/1.00	0.00/0.94	0.31/0.26	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 12	0.00/0.15	0.00/1.00	0.00/0.94	0.31/0.31	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 13	0.00/0.15	0.00/1.00	0.00/0.94	0.31/0.22	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 14	0.00/0.15	0.00/1.00	0.04/0.94	0.31/0.14	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 15	0.00/0.15	0.00/1.00	0.04/0.94	0.31/0.14	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 16	0.13/0.15	0.47/1.00	0.51/0.94	0.31/0.14	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 17	0.00/0.15	0.00/1.00	0.04/0.94	0.31/0.14	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 18	0.00/0.15	0.00/1.00	0.04/0.94	0.31/0.14	0.38/0.41	1.00/0.59	0.75/0.25	0.12/0.09
min. 19	0.00/0.15	0.00/1.00	0.04/0.94	0.31/0.14	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 20	0.09/0.15	0.21/1.00	0.28/0.94	0.31/0.00	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 21	0.09/0.15	0.21/1.00	0.28/0.94	0.31/0.00	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 22	0.09/0.15	0.21/1.00	0.28/0.94	0.31/0.00	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 23	0.09/0.15	0.21/1.00	0.28/0.94	0.31/0.00	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 24	0.09/0.15	0.21/1.00	0.28/0.94	0.31/0.00	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 25	0.13/0.15	0.47/1.00	0.51/0.94	0.31/0.14	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 26	0.09/0.15	0.21/1.00	0.28/0.94	0.31/0.00	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 27	0.00/0.15	0.00/1.00	0.04/0.94	0.31/0.14	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 28	0.00/0.15	0.00/1.00	0.00/0.94	0.31/0.31	0.31/0.41	1.00/0.88	0.75/0.47	0.12/0.11
min. 29	0.00/0.15	0.00/1.00	0.00/0.94	0.31/0.22	0.38/0.41	1.00/0.59	0.75/0.25	0.12/0.09
min. 30	0.00/0.15	0.00/1.00	0.00/0.94	0.31/0.30				

Bibliography

Andrés, C., Llana, L. y Núñez, M. Self-adaptive fuzzy-timed systems. En *13th IEEE Congress on Evolutionary Computation, CEC'11*, páginas 115–122. IEEE Computer Society, 2011.

Bjørner, D. y Havelund, K. 40 years of Formal Methods - some obstacles and some possibilities? En *19th Int. Symposium on Formal Methods in the Development of Computing Systems, FM'15, LNCS 8442*, páginas 42–61. Springer, 2014.

Boubeta-Puig, J., Bravetti, M., Llana, L. y Merayo, M. G. Analysis of temporal complex events in sensor networks. *Journal of Information and Telecommunication*, vol. 1(3), páginas 273–289, 2017a.

Boubeta-Puig, J., Camacho, A., Llana, L. y Núñez, M. A formal framework to specify and test systems with fuzzy-time information. En *14th Int. Work-Conf. on Artificial Neural Networks, IWANN'17, LNCS 10306*, páginas 403–414. Springer, 2017b.

Boulanger, J.-L., editor. *Industrial Use of Formal Methods: Formal Verification*. Wiley, 2013.

Calvo, I., Merayo, M. G. y Núñez, M. An improved and tool-supported fuzzy automata framework to analyze heart data. En *10th Asian Conference on Intelligent Information and Database Systems, ACIIDS'18, LNAI 10751*, páginas 694–704. Springer, 2018a.

Calvo, I., Merayo, M. G. y Núñez, M. `AUNTY`: A tool to automatically analyze data using fuzzy automata. En *3rd Int. Conf. on Computational Intelligence and Applications, ICCIA'18*. IEEE Computer Society, 2018b.

Camacho, A., Merayo, M. G. y Núñez, M. Using fuzzy automata to diagnose and predict heart problems. En *19th IEEE Congress on Evolu-*

tionary Computation, CEC'17, páginas 846–853. IEEE Computer Society, 2017.

CHEN, T., DICIOLLA, M., KWIATKOWSKA, M. y MEREACRE, A. Quantitative verification of implantable cardiac pacemakers over hybrid heart models. *Information and Computation*, vol. 236, páginas 87–101, 2014.

DEJANOVIĆ, I., VADERNA, R., MILOSAVLJEVIĆ, G. y VUKOVIĆ, V. TextX: A Python tool for Domain-Specific Languages implementation. *Knowledge-Based Systems*, vol. 115, páginas 1–4, 2017.

DOOSTFATEMEH, M. y KREMER, S. C. New directions in fuzzy automata. *International Journal of Approximate Reasoning*, vol. 38(2), páginas 175–214, 2005.

DORN, G. The fuzzy logic of physiological cardiac hypertrophy. *Hypertension*, vol. 49(5), páginas 962–970, 2007.

GOGOLLA, M. Benefits and problems of formal methods. En *9th Ada-Europe Int. Conf. on Reliable Software Technologies, Ada-Europe'04, LNCS 3063*, páginas 1–15. Springer, 2004.

HAARMARK, C., GRAFF, C., ANDERSEN, M., HARDAHL, T., STRUIJK, J., TOFT, E., XUE, J., ROWLANDSON, G., HANSEN, P. y KANTERS, J. Reference values of electrocardiogram repolarization variables in a healthy population. *Journal of electrocardiology*, vol. 43(1), páginas 31–39, 2010.

HOZO, S., DJULBEGOVIC, B. y HOZO, I. Estimating the mean and variance from the median, range, and the size of a sample. *BMC medical research methodology*, vol. 5(1), página 1, 2005.

HUNTER, P., PULLAN, A. y SMAILL, B. Modeling total heart function. *Annual review of biomedical engineering*, vol. 5(1), páginas 147–177, 2003.

JIANG, Z., A.CONNOLLY y MANGHARAM, R. Using the virtual heart model to validate the mode-switch pacemaker operation. En *32nd Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society, EMBC'10*, páginas 6690–6693. IEEE Computer Society, 2010.

KHACHATURIAN, Z., KERR, J., KRUGER, R. y SCHACHTER, J. A methodological note: Comparison between period and rate data in studies of cardiac function. *Psychophysiology*, vol. 9(5), páginas 539–545, 1972.

LAMPORT, L. Who builds a house without drawing blueprints? *Communications of the ACM*, vol. 58(4), páginas 38–41, 2015.

MAGNANI, L. y BERTOLOTTI, T., editores. *Handbook of Model-Based Science*. Springer, 2017.

MÉRY, D. y SINGH, N. Formalization of heart models based on the conduction of electrical impulses and cellular automata. En *1st Int. Symp. on Foundations of Health Informatics Engineering and Systems, FHIES'11, LNCS 7151*. Springer, 2012.

MOODY, G. y MARK, R. The impact of the MIT-BIH arrhythmia database. *IEEE Engineering in Medicine and Biology*, vol. 20(3), páginas 45–50, 2001.

MOODY, G. B. RR intervals, heart rate and HRV Howto. `https://www.physionet.org/tutorials/hrv/`, 2016.

MORDESON, J. N. y MALIK, D. S. *Fuzzy automata and languages: theory and applications*. Chapman & Hall/CRC, 2002.

MRAZ, M., LAPANJA, I., ZIMIC, N. y VIRANT, J. Fuzzy numbers as inputs to fuzzy automata. En *18th Int. Conf. of the North American Fuzzy Information Processing Society, NAFIPS'99*, páginas 453–456. IEEE Computer Society, 1999.

NEWCOMBE, C., RATH, T., ZHANG, F., MUNTEANU, B., BROOKER, M. y DEARDEUFF, M. How Amazon web services uses formal methods. *Communications of the ACM*, vol. 58(4), páginas 66–73, 2015.

RIJNBEEK, P., VAN HERPEN, G., BOTS, M., MAN, S., VERWEIJ, N., HOFMAN, A., HILLEGE, H., NUMANS, M., SWENNE, C., WITTEMAN, J. ET AL. Normal values of the electrocardiogram for ages 16–90 years. *Journal of electrocardiology*, vol. 47(6), páginas 914–921, 2014.

WEE, W. G. y FU, K. S. A formulation of fuzzy automata and its application as a model of learning systems. *IEEE Transactions on Systems Science and Cybernetics*, vol. 5(3), páginas 215–223, 1969.

WOJTASIK, A., JAWORSKI, Z., KUŹMICZ, W., WIELGUS, A., WAŁKANIS, A. y SARNA, D. Fuzzy logic controller for rate-adaptive heart pacemaker. *Applied Soft Computing*, vol. 4(3), páginas 259–270, 2004.

ZADEH, L. A. Fuzzy sets. *Information and Control*, vol. 8(3), páginas 338–353, 1965.

ZADEH, L. A. *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems*. Advances in Fuzzy Systems - Applications and Theory: vol. 6. World Scientific Press, 1996.

List of acronyms

BNF: Backus-Naur form.

BPM: heartbeats per minute.

csv: comma-separated values.

ECG: electrocardiogram.

GoC: grade of confidence.

RR interval: interval between two consecutive R waves.